REMARKS

Claims 1, 4-12, 31-39, 42-44, 46-51, 54-56, and 58-62 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 1, 4-12, 31-39, 42-44, 46-51, 54-56 and 58-62 under 35 U.S.C. § 103(a) as being unpatentable over "The Error Handling Interface (H5E)" (hereinafter "H5E-A") in view of Kouznetsov et al. (U.S. Publication 2004/0010703) (hereinafter ""Kouznetsov"). Applicants respectfully traverse this rejection for at least the following reasons.

In regard to claim 1, the cited art does not teach or suggest ***in each of two or more threads of a multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in a memory storage area private to the thread in accordance with an application programming interface (API) to an error trace mechanism.*** As illustrated in Fig. 2 and described in the accompanying description (e.g., paragraph [0021]) of Applicant's specification, each of the two or more threads in the multithreaded program is associated with a particular, separate and distinct, "thread private data", or "private data area", for the thread, which is exemplary of a memory storage area private to the corresponding thread. Error trace elements generated for a particular thread are stored to the corresponding private data area, a memory storage area private to the thread, as is clearly illustrated in Fig. 2 and disclosed in paragraph [0021]:

> Each thread private data 100 is a storage area (memory) specific to a particular thread. Each thread may store and access data in its associated thread private data 100 for the thread. In one embodiment, when error traces are stored using thread private data, all error trace elements generated for a particular thread are recorded in the thread's private data area. Thus, each thread private data 100 may store error traces 104, if any, for its associated thread.

The H5E-A reference does not teach or suggest an error trace mechanism for a multithreaded program configured to, for each error generated by one or more functions executed in each of **two or more threads** in the multithreaded program, store an error trace element in a memory storage area **private to** the corresponding thread, as is recited in claim 1. Further, the H5E-A reference does not teach or suggest an error trace mechanism configured to obtain an error trace for each of **two or more threads** of a multithreaded program, wherein each error trace includes one or more error trace elements specific to the corresponding thread, as is recited in claim 1.

The Examiner asserts Kouznetsov discloses "two or more threads of a multithreaded program and storing an error trace element in a memory area private to the thread for each of the two or more threads," citing paragraph [0408]. Paragraph [0407]-[0408] read (emphasis added):

> [0407] Dynamic Memory API
> [0408] The abstraction library provides two sets of dynamic memory allocation methods for multi-threaded applications. Under operating systems like Unix/Linix and Windows NT (Win32), a block of memory allocated from one thread using the malloc( ) function can be de-allocated in another thread, but under systems like Symbian 6.x. **memory de-allocation** must be preformed by the thread that created it. The AlMemSAlloc and AlMemSFree functions are provided to allocate/deallocate dynamic memory from the process thread, and the AlMemAlloc and the AlMemFree functions are provided for thread-specific memory allocation/deallocation. Under Unix/Linux and Win32, both AlMemSAlloc and AlMemAlloc are mapped to the POSIX malloc. See Table 81.

Applicants note that the portion of Kouznetsov relied upon by the Examiner is from Appendix A, which describes Kouznetsov's "abstract library" or "wireless abstraction library." (See, e.g., [0397]-[0399]). In paragraph [0399], Kouznetsov states "The design goal of the wireless abstraction library is to shield the application from platform-specific implementation details while providing consistent interfaces to the following subsystems in Table 75." Table 75 recites dynamic memory allocation, process enumeration and control, threading and thread synchronization, storage I/O, network socket I/O, string conversion, and system event handling as the subsystems to which interfaces are provided.

Paragraph [0408] simply teaches an API to dynamic memory allocation and deallocation functions for multi-threaded applications that is part of Kouznetsov's "abstract library" or "wireless abstraction library", and that in some operating systems a block of memory allocated from one thread can only be deallocated by that thread. While the citation teaches that, in some operating systems, a block of memory allocated from one thread can only be deallocated by that thread, the citation <u>does not teach</u> the notion of a block of memory that is <u>**private** to the thread</u> as recited in Applicants' claim. Kouznetsov only teaches that, in some operating systems, a block of memory allocated from one thread can only be <u>deallocated</u> by that thread; Kouznetsov does <u>not</u> teach that a block of memory so allocated can only be <u>accessed</u> by that thread. In other words, **the citation from Kouznetsov does <u>not</u> teach the notion of a memory storage area <u>private to the thread</u>.** As previously noted, in Applicants' claim each of the two or more threads in the multithreaded program is associated with a particular, separate and distinct, "thread <u>private</u> data", or "<u>private</u> data area", for the thread, which is a memory storage area <u>private to</u> the corresponding thread. Nor is the notion of a memory storage area <u>private to</u> a corresponding thread taught elsewhere in the Kouznetsov reference. Furthermore, it is noted that the modifier "thread-specific" in the citation from Kouznetsov clearly refers to the memory <u>allocation/deallocation processes or methods</u>, and is not directly used as a modifier for the "memory" itself. In addition, "thread-<u>specific</u>" clearly does not mean the same thing as "thread <u>private</u>" or "private to the thread".

Furthermore, the paragraph [0408] citation clearly does not teach, nor does Kouznetsov elsewhere teach, the notion of **<u>storing an error trace element</u> in a memory storage area private to the thread**. Moreover, claim 1 recites that such an error trace element is stored in the memory storage area private to the thread <u>for each error generated by one or more functions executed in the thread</u>. The Kouznetsov reference is completely silent as to these limitations as recited in claim 1 when viewed as a whole.

All that Applicants can find in Kouznetsov in regard to errors related to threads is paragraphs [0401]-0402], which describe an aspect of Kouznetsov's "wireless abstraction library" (Applicants' comment added in []):

> [0401] Error Functions
> [0402] The abstraction library provides two functions to set and retrieve the last-error code for the calling thread. See Table 77. [Kouznetsov appears to mean Table 78, which follows:]
>
> TABLE 78
> Error API
>
> | Name | Arguments | Returns | Description |
> |---|---|---|---|
> | AlErrGetLast | void | AlErrorCode | retrieves last-error code |
> | AlErrSetLast | AlErrorCode | void | sets the last-error code |

This citation from the Kouznetsov reference clearly does not teach or suggest the above limitations as recited in claim 1, nor are the above limitations taught or suggested elsewhere in Kouznetsov.

Furthermore, the Examiner has not stated a proper reason to combine the teachings of the cited art. The Examiner asserts "it would have been obvious…to incorporate the teachings of Kouznetsov into the teachings of H5E-A to include multithreaded support and storing error trace elements in a memory area private to the thread" because one of ordinary skill in the art would "want to provide support for system such as Symbia[n 6.x] as suggested by Kouznetsov." As previously noted, the portion of Kouznetsov cited by the Examiner only teaches that "the abstraction library provides two sets of dynamic memory allocation methods for multi-threaded applications" and that in some operating systems "memory de-allocation must be preformed by the thread that created it", and does not teach the limitations that the Examiner asserts it teaches. Furthermore, Kouznetsov clearly does not teach "multithreaded support" that would be combinable with H5E-A to produce anything like what is recited in Applicants' claim 1. In addition, contrary to the Examiner's assertion, Kouznetsov clearly does not teach "storing error trace elements in a memory area private to the thread." Furthermore, the portion of Kouznetsov relied upon by the Examiner is from Appendix A, which describes Kouznetsov's "abstract library" or "<u>wireless abstraction library</u>." H5E is but one interface or API of the Hierarchical Data Format 5 (HDF5) architecture. HDF5 and Kouznetsov's "wireless abstraction library" are distinctly different technologies targeted at distinctly different areas. It is not at all clear that combining Kouznetsov's "wireless abstraction library" with HDF5/H5E is even possible, or if possible that there would be any reason to combine such disparate technologies. Thus, the Examiner's assertion that "it would have been obvious…to incorporate the teachings of Kouznetsov into the teachings of H5E-A" to "provide support for systems such as Symbian 6.x" is not supported by the evidence of record and is found only in hindsight. In any case, simply adding Kouznetsov's memory allocation and deallocation functions as described in paragraph [0408] to H5E-A, even if such a combination were possible, would not produce anything like what is recited in Applicants' claim 1.

Moreover, simply adding Kouznetsov's memory allocation and deallocation functions to H5E-A, even if such a combination were possible, would not even achieve the goal asserted by the Examiner of "providing support for systems such as Symbian 6.x." As noted above, H5E is but one API of the Hierarchical Data Format 5 (HDF5) architecture. HDF5 and Symbian 6.x are distinctly different technologies targeted at distinctly different areas. It is not at all clear that the notion of "providing support for systems such as Symbian 6.x" in HDF5 or more specifically in the H5E library of HDF5 is even possible, or if possible that it would be desirable. For that matter, it's not certain that such a notion is even meaningful. Thus, the Examiner's assertion that "one of ordinary skill in the art [would] want to provide support for system such as Symbian 6.x" in a library such as H5E or more generally in an architecture such as HDF5 is not supported by the evidence of record and is found only in hindsight.

Thus, one of ordinary skill would not have combined the teachings of Kouznetsov with the teachings of H5E-A in the manner proposed by the Examiner. Accordingly, the Examiner has failed to establish a *prima facie* case of obviousness.

Furthermore, there is no enabling description in the cited art for a multithreaded embodiment of the teachings of the H5E-A that would be achieved by such a combination, if the combination were possible. The references do not describe how a multi-threaded version of the teachings of H5E-A would work, or in any way provided an enabling disclosure for a multi-threaded version of the teachings of H5E-A. "In order to render a claimed apparatus or method obvious, the prior art must enable one skilled in the art to make and use the apparatus or method." *Motorola, Inc. v. Interdigital Tech. Corp.*, 43 USPQ2d 1481, 1489 (Fed. Cir. 1997) (quoting *Beckman Instruments, Inc. v. LKB Produkter AB*, 13 USPQ2d 1301, 1304 (Fed. Cir. 1989)); see also *Rockwell Int'l Corp. v. United States*, 47 USPQ2d 1027, 1032 (Fed. Cir. 1998). Since the cited art does not enable a multi-threaded version of its teachings, the rejection is improper.

Thus, for at least the reasons presented above, the rejection of claim 1 is not supported by the cited prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 1 also apply to claims 9, 31, 36, 39, 47, 51, and 59.

Furthermore, the Examiner has failed to state a *prima facie* rejection for claims 47 and 59. The Examiner only states in regard to claim 47 to "see reason of rejection of claim 1", and rejects claim 59 "for the same reason set forth in connection of the rejection of claim 47". *However, claims 47 and 39 have different scope than claim 1.* **Since the Examiner has failed to address the differences between the claims, the Examiner's rejection of claims 47 and 59 is additionally improper.**

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-69401/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: August 14, 2008